The Design and Implementation of a Smartphone Illuminance Meter

by

Devon D. Sparks

SUBMITTED TO THE DEPARTMENT OF ARCHITECTURE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN ARCHITECTURE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Febuary 2013

Signature of Author.....

Department of Architecture
January 18, 2013

Certified by.....

Christoph Reinhart, PhD
Associate Professor of Building Technology

Accepted by.....

J. Meejin Yoon, MAUD
Associate Professor of Architecture
Director of the Undergraduate Architecture Program

The Design and Implementation of a Smartphone Illuminance Meter

by

Devon D. Sparks

Submitted to the Department of Architecture on January 18th, 2013
in partial fulfillment of the requirements of the Degree of
Bachelor of Science in Architecture

Abstract

The proliferation of consumer smartphones has offered new opportunities for environmental sensing and mobile computation. Recent smartphone models – equipped with GPS trackers, accelerometers, megapixel cameras and rich software stacks – offer the possibility of emulating specialized tools completely in software. This paper documents recent efforts to build a robust, smartphone-based illuminance meter application, and describes its prototype implementation. Though hardware and software constraints prevent the complete development of such a prototype, its use and potential are demonstrated.

Thesis Supervisor: Christoph Reinhart

Title: Associate Professor of Building Technology

The Design and Implementation of a Smartphone Illuminance Meter

Devon D. Sparks

January 2013

1 Introduction

Rapid advances in consumer electronics and advertising have transformed the mobile device market. Single-function mobile phones of the early 1990s were overtaken by "feature phones", only to be usurped by the powerful, touchscreen smartphones of the last few years. Now standard issue for 74% of Americans aged 24-35 [15], smartphones have been ceaselesly recast as tour guides, news readers, desktop calculators, event planners, and sound mixers. When one machine has the ability become most others with the swipe of a finger, there are two practical issues to consider: what is the upper bound on a smartphone's ability to replace specialized tools, and how might this be changed? This paper explores both questions in the domain of building science, using a smartphone-based illuminance meter as its test application. Its conclusions are straightforward: that despite their power, existing smartphones are not yet fit-for-purpose as digital light meters, and that some of their failings can be fixed. It examines the limitations of current smartphone software stacks in supporting accurate illuminance calculation, and speculates on the potential use of smartphones as tools for teaching building science and architectural daylighting.

2 Review of Existing Smartphone Light Meters

The promise of smartphones as tools for environmental monitoring has been widely published [9] [12]. Illuminance meter applications for two leading smartphone software stacks – Google's Android and Apple's iOS – make

Name	Developer	Platform	Rating
1-Ap Light Sensor	1-Ap	Android	2.8/5.0, 49 Reviews
Light meter	Manas Gajare	Android	1.0/5.0, 3 Reviews
Lightmeter	MikLab	Android	2.7/5.0, 76 Reviews
MobiLux	Tech.Music-Bee	Android	2.6/5.0, 86 Reviews
Light Lux Meter	Herbert Mhlburger	Android	3.1/5.0, 27 Reviews
LuxMobile	NECTECH	Android	3.1/5.0, 17 Reviews
Light Meter	Borce Trajkovski	Android	3.4/5.0, 341 Reviews
Lux Meter	oxdb.net	Android	2.5/5.0, 158 Reviews
Light Meter	Roger Yang	Android	3.2/5.0, 29 Reviews
LuxMeter	App Manufactory	iOS	4 Stars, 1 Review
Check The Light	Stichting	iOS	1 Star, 5 Reviews
LightMeter	whitegoods	iOS	3 Stars, 12 Reviews

Table 1: Ratings of Reviewed Smartphone Light Meters, October 2012

use of a smartphone's internal light sensor to estimate scene illuminance. User reviews are consistently poor (see Table 1), due in large part to the limitations of the phone's ambient light sensor. Though a photodiode may be an appropriate tool for adjusting an LCD backlight, its sensitivity is too gross for accurate illuminance calculation. Of 12 reviewed illuminace meters available in Google Play and Apple App Store, all depend on the limited accuracy of the phone's ambient light sensor. Fortunately, the large number of disappointed reviews demonstrates the clear interest in a more robust smartphone light meter, and the release of increasingly sophisticated software stacks lays fertile ground for its development.

3 Proposal for a New Light Meter

A proper smartphone light meter should embody the basic qualities of a specialized meter (small size, easy of use), but leverage the processing power and network connectivity of mobile phones. To accurately measure scene illuminance, it must use a fine sensor, recover spot luminance from encoded pixel values, and give results in reasonable time for side- and toplit spaces. These requirements demand that illuminance calculation be converted into a multi-stage pipeline: capture, merging, and calculation (see Figure 1). Each module, along with its prototype implementation built on a Samsung Galaxy

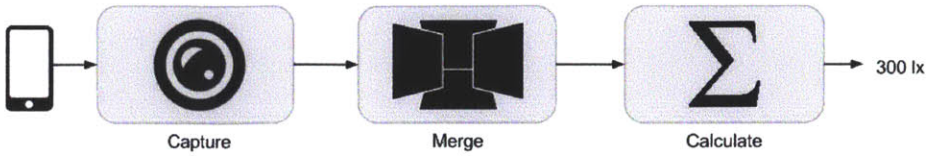


Figure 1: Illuminance Calculation Workflow

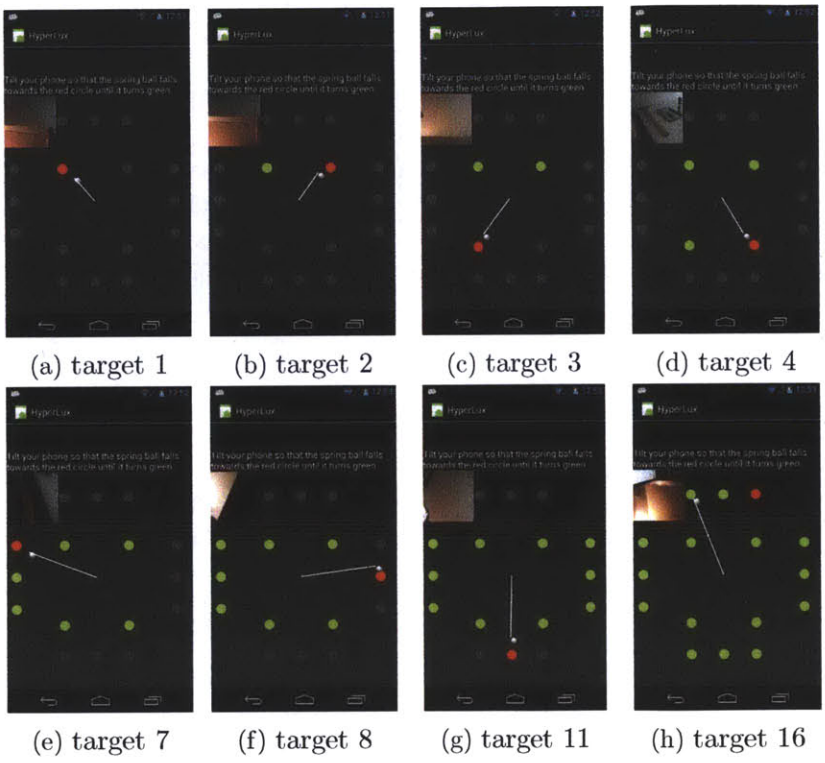
Nexus running Android 4.1, will be described in the following sections.

3.1 The Capture Pipeline

To accurately estimate scene illuminance, a smartphone light meter must have a field of view large enough to account for top and side lighting. As most smartphone cameras (including the Galaxy Nexus) have a relatively narrow field of view (50-60 degrees), multiple images must be captured to allow for panorama generation. To guide the image capture process, a new interface is needed.

In the prototype implementation, the user is presented with a "mass and spring" simulation, controlled by the phone's orientation sensors and a software-based Verlet path integrator. As the phone shifts orientation, a virtual mass stretches away from the screen's center proportional to the tilt (see Figure 2). Once in contact with the active on-screen target (shown as a small red circle in the prototype interface), the phone's front-facing camera captures several lossy (i.e. JPEG encoded) images of the current scene – ideally with varying exposure times – and writes them to internal storage for later processing. The targets are positioned to maximize the chance of feature matching between image pairs. As the simulation is rotationally invariant, it is assumed the user remains stationary throughout the capture exercise.

The intent behind a mass-and-spring simulation is twofold: first, to force the user to slow his movement, and focus his attention, so that crisp scene fragments can be captured; and second, to reduce his perception of elapsed time while background threads complete image analysis. If illuminance calculations must be computationally intensive, the best an application designer can do is make the wait an imperceptible one.



(a) target 1 (b) target 2 (c) target 3 (d) target 4

(e) target 7 (f) target 8 (g) target 11 (h) target 16

Figure 2: Image Capture Interface Prototype

3.2 The Merging Pipeline

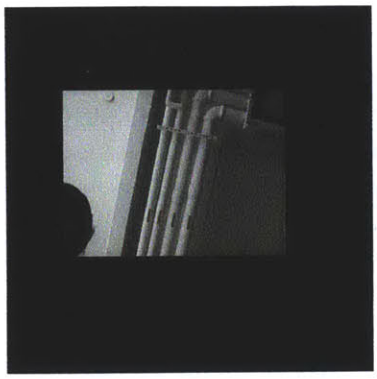
Once captured, the application aligns images pairwise, storing homographies calculated using the algorithms of BoofCV [7], a pure-Java implementation of OpenCV [5] (see Figure 3). It then attempts to collapse these images into a single mosaic with a 90°effective field of view. These images become the input for the next pipeline stage, which implements fisheye projection using a cubic environment mapper [11].

Generating large image mosaics on a small phone is not easy. It is computationally expensive (both in time and space), especially on a machine with a 16-24MB memory limit per application[1]. Fortunately, experiments with the prototype image stitching module were largely successful, generating complete, 90°field of view planar images in 20-30 seconds each in more than 75% of trials.

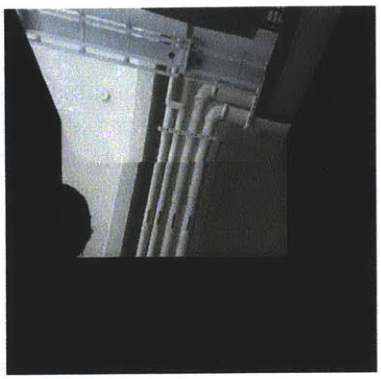
However, there are strong caveats to this approach. Fundamental properties of the workflow that make it a poor match for smartphones. Successful feature matching of the overhead scene required 8-12 source images, pairwise homography calculation, and repeated reading from external storage to avoid Android's internal memory constraints. Homography calculation depends on the success of feature correspondence between overlapping images, a challenge if the overhead scene has few distinguishing features. In order to generate a full 180°fisheye perspective, four additional mosaics must be generated, each representing the upper half of the user's view in each of the cardinal directions. This says nothing of high dynamic range (HDR) image generation or response curve recovery, described in the next section.

One research-grade mobile software stack has managed to mitigate some of these challenges [8], at the cost of specific smartphone hardware, a custom operating system installation, and modified firmware. Other efforts to perform multi-directional HDR panorama generation have either offloaded the feature correspondence calculations to back-end servers, or used one of the two image registration routines outlined in Reinhard, Ward et. al. [14, p. 122]. The faster of these routines – *the mean threshold bitmap alignment routine* – is unsuited for images taken at different view angles, making it a

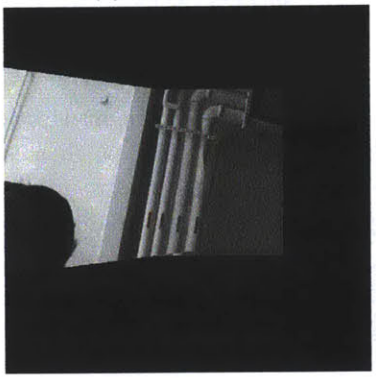
[1]See the section *Displaying Bitmaps Efficiently* of the Android Training Manual for details: http://developer.android.com/training/displaying-bitmaps/index.html. There is some discussion that this hard limit can be overcome through use of Google's Native Development Kit (NDK), though this option was not pursued here; tight resource constraints are not the only count against a smartphone-based illuminance meter.



(a) Central Image



(b) Top-Central Mosaic



(c) Left-Central Mosaic

(d) Right-Central Mosaic

(e) Composite Mosaic

Figure 3: Sample Mosaic Generation

poor match for smartphone deployment.

Perhaps the greatest threat to accurate illuminance calculation using image mosaic techniques is the introduction of object distortion during merging. In Figure 3e, for example, small discrepancies in feature correspondence between successive image sets created an erroneous fragment of the author's head in the lower-left corner of the image. These erroneous fragments (a consequence of the stochastic nature of feature matching routines) are troublesome for illuminance calculations when they under- or overrepresent the size of light sources. This problem can be overcome (at additional cost and loss of convenience) with the use of an external fisheye lens.

3.3 The Calculation Pipeline

Under the assumption that an HDR image mosaic can be generated from scene fragments, implementation of a luminance integral – the central calculation in a smartphone light meter – is trivial. Given a smartphone camera's unique response curve, integration involves a linear transformation, lookup, and Reimann sum for all pixels in the input image. In the prototype implementation, luminance integration on sample, low dynamic range images never took more than a few seconds.

While outputting a single illuminance value would fulfill all the requirements of a fit-for-purpose light meter, the high resolution display and network connectivity of smartphones offers new opportunities. With user permission, illuminance calculations could be uploaded to an online database, capable of archiving HDR images and GPS coordinates along with a qualitative description of the scene. Such an archive could become the basis of further work in the human factors associated with architectural daylighting, and improve student intuition of lighting fundamentals. Such collaborative tools have been deployed in other fields (as in the SoundAroundYou.com environmental audio archive from the University of Salford), and there's little reason to think their success could not be replicated by daylighting practioners and students.

4 Hardware Limitations of Smartphones

Methods for computing scene illuminance using mass-market digital cameras are well known [16]. Though a smartphone camera has no mechan-

ically adjustable aperture, its sensor array can transform the number of photons impinging each sensor into a proportional voltage, perceived as a pixel brightness in the final image. Many smartphone cameras now support multi-megapixel sensors, but remain bound to software APIs that encode 24-bit image formats. One byte for each red, green and blue color channel allows for 16.7 million colors per pixel, but only 256 values per channel. In scenes with some combination of sunlighting ($10^5 cd/m^2$) and artificial lighting ($10^2 cd/m^2$), one byte per channel per pixel can never represent the full dynamic range in a single image; most snapshots will be under- or overexposed. However, capturing multiple images of a static scene with varying exposure times provides an array of RGB values for each pixel, at least one of which will be properly exposed. By choosing proper exposures on a pixel-by-pixel basis, an HDR image can be generated that removes the limitations of standard, low dynamic range (LDR) imaging.

If accurately captured, each pixel in a source image can then undergo a change of basis from RGB into the CIE XYZ color space [13], where the CIE Y value closely matches spectral luminous efficiency of the human eye for phototopic vision [1]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Assuming an ideal, linear image sensor, each spot luminance can be normalized by dividing out its exposure time [14, p. 117]. As a practical matter, most digital cameras do not have a linear response; manufacturers often introduce gamma correction into their image processing pipelines in order to generate more pleasing images for monitors and print. Restoring this linear mapping requires recovery of the camera's unique response curve, called the Opto-Electronic Conversion Function (OECF), and described in International Standard ISO 14524 [2]. Thankfully, once captured and stored, the camera response curve may can be used for subsequent exposures.

One popular algorithm for recovering this mapping, due to Debevec and Malik [10], leverages the fact that by capturing different exposures of a static scene, "one is effectively sampling the camera response function at each pixel." [14, p. 137] When the exposure values are plotted against each pixel value, Debevec and Malik use linear optimization to find a smooth curve that minimizes the mean-squared error over each of the three red, green, and blue color channels. If the camera hardware performs automatic and proprietary on-chip image processing, linear optimization on one or

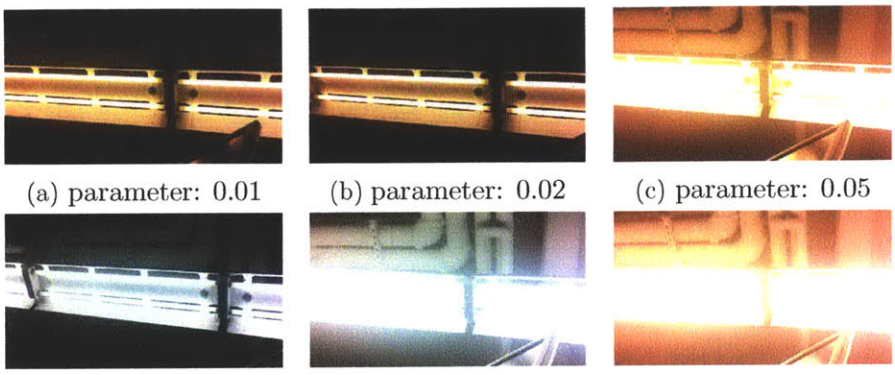
more channels may fail, and an accurate response curve will not be found.

Response recovery on a Samsung Galaxy Nexus, a Google-endorsed smartphone running Android 4.1, proved unsuccessful for just this reason. Anywhere Software's *Photosphere* failed to produce an accurate response curve on all of seven independent trials, each using seven input images taken with Android's built-in camera application (one for each of the four available *capture modes*, and three additional images using the built-in *area metering* feature). Few image sets converged at all, and those that did gave polynomials of the first or second degree – less accurate than a generic sRGB response curve. Response recovery on images taken with the manual exposure settings of a Canon EOS Rebel T3, in contrast, were consistently successful, implying that the lack of manual exposure control and on-chip image processing of the Android-based smartphone was stunting response recovery.

Existing Android camera parameters – like area metering, and exposure compensation – are not appropriate substitutes for manual exposure: the former for its inability to reliably alter the automatic exposure settings of the camera firmware, and the latter for its application as a post-processing step, after pixel values have been encoded. Apple's iOS camera APIs, which provide similar features, have similar shortcomings. Use of a generic response curve is also ill-fated, as Reinhard, Ward et. al. note [14, p. 136]:

> "Assuming an sRGB response curve is unwise, because most makers boost image contrast beyond the standard sRGB gamma to produce a livelier image. There is often some modification as well at the ends of the curves, to provide software highlights and reduce noise visibility in shadows."

It is a surprising fact that, at the time of writing, neither Google Android nor Apple's iOS provide official support for manual exposure control – requisite for HDR generation, response curve recovery, and accurate illuminance calculation. This restriction is well documented for the Android platform in Google Android Issue #5692 [3], and on iOS in the *AVCaptureDevice* Class Reference [4]. Given that illuminance calculation with mass-market digital cameras is well understood [16], it is clear this restriction affects smartphone camera hardware and software stacks specifically.



(a) parameter: 0.01 (b) parameter: 0.02 (c) parameter: 0.05

(d) parameter: 1.0 (e) parameter: 10.0 (f) parameter: 100.0

Figure 4: Apple iOS Manual Exposure API Results

5 Smartphone Camera API Restrictions

As of late 2012, the iOS and Android software stacks provide no official support for manual exposure control for smartphone cameras. Beginning with iOS 6.0, however, HDR image generation is possible within Apple's built-in camera application. A review of the *AVCaptureDevice* class headers in iOS 6.0 reveals several private method calls for manual exposure, including one **setExposureDuration:** that takes a single C structure as an argument. Unfortunately, the class headers can only provide information on argument *types* required by this method, and can say nothing about their *meaning*. Calling the **setManualExposureSupportEnabled:** method of an *AVCaptureDevice* with **YES** enables the new exposure mode. Manual exposure on an iPhone camera can then be enabled by passing the new mode flag (3) to the *AVCaptureDevice*.

Brute-force experimentation with the structure arguments to **setExposureDuration:** produced the image set in Figure 4 on an iPhone 4 with iOS 6.0, suggesting that a more flexible (if currently undocumented) manual exposure camera API may exist for Apple iOS devices.

As this paper went to print, Google Inc. released Android OS 4.2, which includes a new HDR scene mode for the built-in camera application. This theoretically provides the same HDR functionality available in iOS 6.0, but requires modifications to the system kernel and camera drivers. The HDR scene mode only supports the Google Nexus 4 hardware at the time of

writing, with the Android 4.2 APIs still unable to provide application-level manual exposure or white balance controls.

6 Discussion

The inability to control the underlying camera hardware directly from the iOS or Android software stack regrettably stall efforts to develop a light meter application suitable for design science teaching. This is both a limitation of existing software stacks and hardware specifications. The well-documented use of mass-market digital cameras and PC-based image analysis workflows implies that, with some potential hardware and software modifications, smartphones could emulate the behavior of existing illuminance meters.

The hardware limitations of smartphone camera modules may be the greatest obstacle to development. Such modules are optimized to minimize space, power and manufacturing requirements, maximize on-chip image processing, and produce pleasing images in low-light conditions using short exposure times. Most modules have integrated lens, increasing the chance of lens flaring. These specifications are entirely antithetical to the needs of camera-based illuminance calculation, including: repeatable, low-noise data (which implies a low, fixed sensitivity); manual white balance control; a low-flare lens system; an unaltered tone curve; RAW data capture for photometry calculations; and a mechanically adjustable aperture. More damning is the lack of metadata documenting transformations applied to each image by the camera module. With no record of image processing performed on input data, and no means to control the module's processing pipeline, the relationship between the raw sensor data and output pixel values is unpredictable.

This does not rule out the possibility of a smartphone-*supported* light meter in the future. In the worst case, the motivated user could modify his smartphone's firmware, install a basic Linux installation on its drive, buy an inexpensive, USB-enabled camera with a fisheye lens, and emulate the HDR generation and illuminance calculation workflow normally performed on workstations. While *technically* possible, this approach is hardly *practical.* Smartphone applications appeal because they are configuration-free, instantly available, and device agnostic. An application that requires an external lens, top-of-the-line hardware, and two-minute wait time fails on all three counts, and has little to offer relative to even the most inexpen-

sive specialized meters. If development of a robust smartphone light meters proves to be an unreasonable challenge, a specialized (but inexpensive and open source) meter could be built using off-the-shelf components (e.g. a Raspberry Pi [6] system-on-a-chip, a USB-enabled camera, and an inexpensive fisheye lens). This would offer a viable alternative to commercial light meters for students of architecture and engineering, and be open for exploration and modification in its own right. The author intends to explore this approach in the coming months.

If mobile software vendors (particularly Google and Apple) introduce new camera APIs that support manual exposure control, flexible, developer-level HDR generation may be viable. Some technical cleverness may be required to circumvent application memory limits, but this is a surmountable challenge. Panoramic stitching of wide-angle scenes – important in spaces with considerable side-lighting – will remain technically feasible, if unwieldy. API changes providing native support for panorama generation may improve this lot, though it remains to be seen whether it will be enough to offset the wait time required between image capture and illuminance calculation. Though it may be a slight kludge, the use of an external, mountable fisheye lens (available for less than the price of some lunches) avoids many of the problems outlined in previous sections. If the goal of a smartphone-based light meter is to develop intuition, and offer first-order accuracy of calculation, it is a boon to avoid computionally expensive image stitching and use a basic lens. Then users, developers, and researchers can focus more on the practical implications of calculation and less of the technical wizardry that made it possible.

References

[1] *ITU-R Recommendation BT.709, Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange.* ITU (International Telecommunications Union), Geneva, 1990.

[2] *ISO 14524.* International Organization for Standardization, 1999.

[3] Android issue #5692. `http://code.google.com/p/android/issues/detail?id=5692`, 2009. 10/10/2012.

[4] Avcapturedevice class reference. `https://developer.apple.com/library/mac/documentation/AVFoundation/Reference/`

AVCaptureDevice_Class/AVCaptureDevice_Class.pdf, 2012. 10/14/2012.

[5] Opencv wiki. http://opencv.willowgarage.com, 2012. 10/12/2012.

[6] Raspberry pi — an arm gnu/linux box for $25. take a byte! http://www.raspberrypi.org, 2012. 12/16/2012.

[7] P. Abeles. Boofcv. http://boofcv.org, 2012. 6/15/2012.

[8] A. Adams, E.-V. Talvala, S. H. Park, D. E. Jacobs, B. Ajdin, N. Gelfand, J. Dolson, D. Vaquero, J. Baek, M. Tico, H. P. A. Lensch, W. Matusik, K. Pulli, M. Horowitz, and M. Levoy. The frankencamera: an experimental platform for computational photography. *ACM Trans. Graph.*, 29(4):29:1–29:12, July 2010.

[9] S. Aram, A. Troiano, and E. Pasero. Environment sensing using smartphone. In *Sensors Applications Symposium (SAS), 2012 IEEE*, pages 1–4. IEEE, 2012.

[10] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. *SIGGRAPH 97*, August 1997.

[11] N. Greene. Environment mapping and other applications of world projections. *Computer Graphics and Applications, IEEE*, 6(11):21 –29, nov. 1986.

[12] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140 –150, September 2010.

[13] C. Poynton. A guided tour of color space. In *SMPTE Advanced Television and Electronic Imaging Conference*, pages 167–180, 1995.

[14] E. Reinhard, G. Ward, S. Pattanaik, and P. Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[15] N. M. Resarch. Young adults and teens lead growth among smartphone owners, September 2012.

[16] D. Wüller and H. Gabele. The usage of digital cameras as illuminance meters. *Electronic Imaging Conference*, 2007.